Refining Bias in Clustering Algorithms via Problem-specific Constraints

Kiri Wagstaff
Department of Computer Science
Cornell University
Ithaca, NY 14853
wkiri@cs.cornell.edu

May 5, 2000

Abstract

Many machine learning techniques have been shown to benefit greatly from the incorporation of background knowledge. This domain-specific knowledge can bridge the gaps left by incomplete data, improve robustness, and provide a directional guide for learning. Ideally, automatic tools could make use of this knowledge and thus not require supervisory tagging. We propose the integration of background knowledge directly into unsupervised learning algorithms. Specifically, we are interested in background knowledge in the form of problem-specific constraints, which allow the automatic exclusion of unpromising areas of the search space. In previous work with a clustering algorithm and instance-level hard constraints, we have demonstrated the promise of this approach, observing improvements in accuracy and efficiency. This document outlines the other kinds of algorithms and constraints we intend to investigate. In addition, we present our intended method for evaluating constraint-based techniques.

1 Introduction

Background knowledge can be a powerful aid to machine learning algorithms. We use this term to refer to any information given to the algorithm that is not contained in the data representation. This information has the potential to provide hints about the true data distribution, improve the detection of outliers or noise in the data, and provide a directional guide for learning. Methods for incorporating knowledge in learning algorithms range from developing problem-specific heuristics to providing an oracle for the algorithm to consult (e.g., De Raedt et al. (1991)). The majority of past work on incorporating background knowledge has focused on supervised methods, which can be applied only when instance-level supervisory information is available. Yet in many cases, this information may be difficult, expensive, or impossible to obtain. In the domain of Natural Language Processing, for example, human annotators are required to provide supervisory information for a variety of tasks, including part of speech tagging, noun phrase coreference, word sense disambiguation, parsing, information extraction (IE), etc. Some of these tasks are particularly time-consuming because the annotations required are more complex than just a class label (e.g., the parse structure of a sentence or a completely filled-in IE template). In other domains, the supervisory information simply is not known. This is the case for many data mining applications where the goal is to detect novel patterns in data (e.g., market basket analysis (Agrawal et al. 1993)). A second example is the research being conducted into protein sequences, where the functions of some proteins are known but others have yet to be discovered (Yona et al. 1999).

We propose the integration of background knowledge directly into clustering algorithms. However, rather than hard-coding general knowledge that must apply over all problems, we are instead interested in augmenting algorithms so that they can make use of problem-specific information. We are specifically interested in background knowledge in the form of problem-specific constraints, which allow the automatic exclusion of unpromising areas of the search space. This merging of information and algorithm is accomplished by a modification of the algorithm's bias.

Work in statistical data analysis has also investigated methods for incorporating some structural information in clustering algorithms. However, these methods have primarily focused on only one type of constraint (contiguity constraints), and as we will show, there are other important kinds of constraints that should also be handled. In addition, most of this research has involved batch algorithms exclusively.

In the next section, we identify what we mean by bias and provide some background on past attempts to modify it in a supervised framework. Next, we review four categories of clustering algorithms and then introduce the concept of using constraints as background knowledge. In Section 5, we outline our previous work with a constraint-clustering algorithm and summarize our findings. Section 6 sets forth our plans for continuing work on the subject, and Section 7 describes methods for evaluation. Finally, we summarize the key research issues involved in this work.

2 Bias

Mitchell (1997) defines *inductive bias* as the set of assumptions that a learner uses to justify its inductive inferences as deductive inferences. In particular, this bias allows the learner to make rational decisions about new data. Formally, Mitchell defines the bias B of a learner L given training data D over testing data X as follows:

$$\forall x_i \in X \,.\, (B \wedge D \wedge x_i) \vdash L(x_i, D)$$

The bias B is the set of all assumptions (actually, any minimal set of assumptions) such that, given the training data D and a new instance x_i, the learner's classification of x_i follows deductively. However, this definition only takes into account the algorithm's behavior with regard to the testing data X. In an unsupervised framework, we are also interested in the algorithm's behavior on the data set D (in some cases, the results of a clustering algorithm may *never* be applied to separate testing data, if the goal is simply to obtain the clusters).

We therefore adopt the broader notion of bias as expressed by Gordon and desJardins (1995), which includes any factor that influences the algorithm's definition or choice of hypotheses to explain D (and optionally, X). Gordon and desJardins identify two classes of bias: *representational* bias, which determines which hypotheses will be considered as potential solutions, and *procedural* bias, which dictates the order in which they will be investigated. We will be concerned primarily with modifications to the representational bias, thereby enabling reductions in the size of the search space.

Algorithms that have a static, hard-wired bias are necessarily limited in their applicability (to problems that conform to that bias). Consequently, researchers have investigated methods for automatically selecting the best algorithm (and therefore bias) for a given problem (Brodley 1994) or dynamically shifting bias as learning proceeds (e.g., Utgoff (1986), Spears and Gordon (1991)). These methods have the benefit of improving the learning rate or of enabling learning that otherwise was not possible (because the original bias was too restrictive). Essentially, these systems have a built-in *meta-bias* which guides their selection of bias.

In general, these methods are concerned with supervised learning tasks; less attention has been devoted to exploring the role of bias in unsupervised learning. As described in Section 1, reducing reliance on supervisory tags or annotations enables the application of learning techniques to many real-world data sets for which little or no such information exists. In many cases, obtaining access to background knowledge will be much easier than labeling each instance in the data set. Therefore, eliminating tags in favor of domain knowledge is a worthwhile goal.

We therefore propose the investigation of problem-specific methods for refining the bias in unsupervised learning algorithms, with the goal of increasing accuracy and improving efficiency. In particular, we will investigate the incorporation of background knowledge in the form of constraints that allow an unsupervised algorithm to automatically avoid unpromising areas of the hypothesis space.

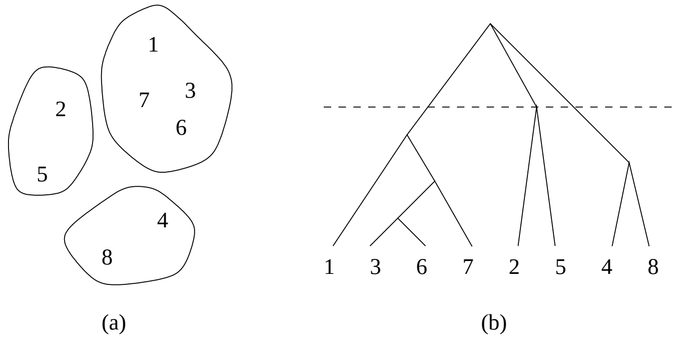


Figure 1: Partition vs. Hierarchy. (a) a three-cluster partition; (b) the corresponding hierarchy

3 Clustering Algorithms

In general, finding the optimal solution to a clustering problem is NP-hard (Křivánek 1991). Anderberg (1973) notes that the number of possible ways to partition n items into k clusters is

$$\mathcal{S}_n^{(k)} = \frac{1}{k!} \sum_{i=0}^{k} (-1)^{k-i} \left(\begin{array}{c} k \\ i \end{array} \right) i^n.$$

This means that, for example, the number of ways to sort 25 items into 5 groups is prohibitively large:

$$\mathcal{S}_{25}^{(5)} = 2,436,684,974,110,751$$

Consequently, much effort has been devoted to developing methods which can reach a reasonably good solution in a reasonable amount of time. A large number of such algorithms have been developed (see, e.g., Hartigan (1975), Anderberg (1973), and Gordon (1981) for compilations). While a comprehensive survey of all clustering algorithms is beyond the scope of this document, we here describe four main categories and provide the details of a representative algorithm for each.

1. **Batch partitioning.** Batch algorithms have access to the entire data set from the start. Batch partition algorithms generally take in a data set D and the desired number of clusters k and return a set of clusters P that *partitions* D, i.e. P is composed of non-overlapping clusters $C_i, i = 1 \ldots k$ such that $\bigcup_{i=1}^{k} C_i = D$. For example, Figure 1(a) shows a partition of eight items into three clusters.

   K-means clustering (MacQueen 1967), also known as *nearest centroid sorting* (Forgy 1965, Jancey 1966) and *iterative relocation*, is the most common batch partitioning algorithm. It works as follows:

   (a) Select k initial cluster centroids, $c_1 \ldots c_k$.
   (b) Assign each instance D_i to its nearest centroid c_j.
   (c) For each cluster, recalculate its centroid based on which instances it contains.
   (d) Alternate between (b) and (c) until convergence.

   This is essentially a search for the best set of k cluster centroids, which also determines the structure of the partition by assigning each instance to its nearest centroid. Starting with some initial configuration, the algorithm iteratively improves its estimate of the cluster centroids until no further changes are possible.

   The alternating nature of the algorithm is reminiscent of EM-based techniques (Dempster et al. 1977), and in fact, different k-means algorithms can be implemented using EM (Mitchell 1997). The Autoclass clustering algorithm (Cheeseman et al. 1988) is a batch partitioning method that makes use of EM methods in a Bayesian framework.

There have been many variants of this basic algorithm. Most involve modifications to step (a); possible ways to select the initial k centroids include using the first k instances in the data set (MacQueen 1967), randomly selecting k instances from the data set, taking any partition of the data set into k clusters and computing the centroids of those clusters (Forgy 1965), etc. MacQueen's version also varies in that it makes only one pass through the data instead of iterating until convergence, thus greatly reducing runtime but at the cost of increased sensitivity to the choice of initial centroids.

Another variation starts with an initial partition and then iteratively selects a pair of instances from different clusters and swaps them, seeking a local maximum according to some objective function.

2. **Batch hierarchical** algorithms take in a data set D and return a hierarchy that reflects structure found in the data set. This hierarchy can be expressed as a series of partitions $P_{1..m}$ of D such that $|P_i| > |P_j|, i < j$. Thus, P_1 is composed of n clusters with each instance in its own cluster, P_m is one cluster that contains all n instances, and all P_j are "nested" in that each cluster in P_j is a subset of some "parent" in P_{j+1}. For example, Figure 1(b) depicts a hierarchy constructed on the same data set as part (a). Each level of the hierarchy represents a partition of the data. The dashed line indicates the level of the tree corresponding to the partition in (a).

The most common batch hierarchical algorithm is the *agglomerative* algorithm (Ward 1963), which proceeds as follows:

(a) Place each instance D_i in its own cluster C_i, forming a partition P_1 of the input D such that $|P_1| = n$. Let $j = 1$.

(b) While $|P_j| > 1$, find the two closest clusters $C_q, C_r \in P_j$. Let $P_{j+1} = P_j \setminus C_q \setminus C_r \bigcup \{C_q \bigcup C_r\}$. Increment j.

Beginning with each instance in its own cluster, the algorithm iteratively merges the two closest clusters.[1] The resulting series of partitions forms a hierarchy (sometimes referred to as a *dendogram*) that was constructed in a bottom-up fashion. This algorithm is step-wise optimal (Gordon 1981).

In order to make use of such an algorithm, we must define what "closest" means when applied to clusters. Generally, we assume that there is some way to measure the distance between two instances, $d(D_i, D_j)$, and then construct a corresponding measure that operates on clusters, $d(C_i, C_j)$. There are three main methods for doing this. First, there are the linkage methods. The *single linkage* method defines

$$d(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y);$$

the *complete linkage* method utilizes

$$d(C_i, C_j) = \max_{x \in C_i, y \in C_j} d(x, y).$$

Second, there are the *centroid-based* methods, for example,

$$d(C_i, C_j) = \sum_{x \in \{C_i \cup C_j\}} d(x, centroid(\{C_i \cup C_j\})).$$

Lastly, there are the *sum-of-squares* methods, for example,

$$d(C_i, C_j) = \sum_{x \in \{C_i \cup C_j\}} d(x, centroid(\{C_i \cup C_j\}))^2.$$

Each notion of distance results in slightly different results. Single-link methods are good at detecting long "chaining" clusters, while complete-link methods are appropriate for hyperspherical clusters. The sum-of-squares methods attempt to reduce within-cluster variance.

[1] Note that this is how the algorithm is usually formulated, but more generally "the two closest clusters" can be replaced with "the two clusters such that $f(P_{j+1})$ is minimized (or maximized)" for some objective function f.

The Lance-Williams update formula (Lance and Williams 1967) is a general form that encompasses all of these alternatives. Assuming that the distance between single-element clusters is the same as the distance between their respective instances, this formula specifies how to update the distance between clusters when two clusters are merged. More precisely, if C_p and C_q are merged, then the distance between the newly merged cluster and any other cluster C_s is given by

$$d(C_p \cup C_q, C_s) = \alpha_1 d(C_p, C_s) + \alpha_2 d(C_q, C_s) + \beta d(C_p, C_q) + \gamma |d(C_p, C_s) - d(C_q, C_s)|$$

Different choices for the parameters α_1, α_2, β, and γ lead to different variations on the general agglomerative algorithm specified above (Gordon 1981).

Less common is the *divisive* algorithm, which works in a top-down fashion by first placing all instances in one cluster, forming the partition P_m, and then iteratively choosing a split point to create a new partition with more constituent clusters. As phrased, this process is much more expensive than the agglomerative method. At each level, rather than selecting a merge point from all possible pairs of the current clusters, one must consider all possible partitions of the current cluster into two sub-clusters. According to Anderberg (1973), the usual simplifying method is *monothetic*, i.e. the algorithm picks one attribute and then finds the best way to partition the instance according to that attribute's values.

3. **Incremental partitioning.** Incremental algorithms are also referred to as *online* algorithms. Rather than having access to all of the instances at once, the algorithm must continually adapt its clustering to an incoming stream of data. Naturally, incremental algorithms are sensitive to the ordering of the data (in some cases, such sensitivity is in fact desired).

   One method is known as *region growing*, where each cluster has a given capacity. The algorithm is as follows:

   (a) For each $D_i \in D$, add D_i to the closest cluster C_j.

   (b) If C_j has reached its maximum capacity, remove it from consideration and continue.

   This somewhat unusual algorithm is appropriate in situations where specifying a cluster capacity makes sense.

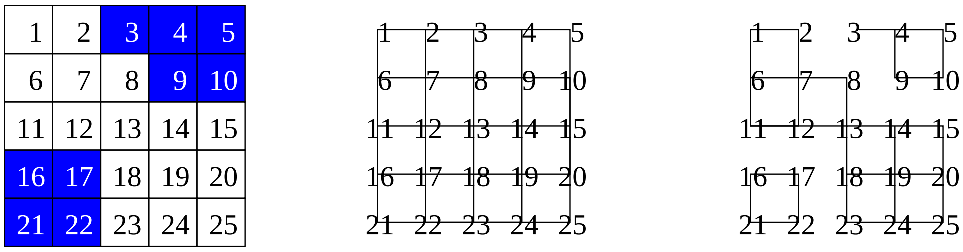
   In addition, incremental partitioning algorithms that are variants of a corresponding incremental hierarchical algorithm exist. For example, we constructed a partitioning version of COBWEB (described in the next section) called COP-COBWEB (see Wagstaff and Cardie (2000) for algorithmic details).

4. **Incremental hierarchical** algorithms also process the input incrementally, but the result is a hierarchy. One example is COBWEB (Fisher 1987). It works as follows:

   (a) For each $D_i \in D$, start at the top level of the existing hierarchy and select the best method for incorporating D_i into that level. The choices are:
      i. **Add** D_i to an existing node.
      ii. Create a **New** node to hold D_i.
      iii. **Merge** two nodes and add D_i to the result.
      iv. **Split** a node and add D_i to one of its children.

   (b) If D_i was placed in a leaf of the hierarchy, stop. Otherwise, recurse on the subtree rooted at the node D_i joined, selecting the best way to incorporate D_i into that subtree.

   This algorithm incrementally builds a hierarchy on the data. However, this hierarchy differs slightly from that presented above in that it need not be a binary tree. In fact, it is common to have several clusters at the top level. The "best" method is selected according to an objective function that attempts to maximize the category utility (Gluck and Corter 1985) of the partition defined by each level of the tree.



(a) (b) (c)

Figure 2: Image Segmentation. (a) the image with pixels numbered; (b) the corresponding contiguity graph of relation R; (c) a three-cluster result after clustering the pixels subject to relation R

4 Background Knowledge as Constraints

Having reviewed a variety of clustering algorithms, we now turn to a discussion of viewing background knowledge as constraints that modify the bias of these algorithms. There is a natural connection between bias in an algorithm and background knowledge. Often, the bias (or meta-bias) hard-coded into an algorithm is chosen due to background knowledge about the class of tasks to be targeted. This permanent bias must then suffice for any problem to which the algorithm is applied. However, for a specific task it is often the case that more precise information is available that can be used to augment the bias in useful ways (AAAI-92 Workshop 1992). In such cases, it is desirable to leverage this background knowledge to refine the algorithmic bias in the proper direction.

In particular, we are interested in improvements that can be obtained with the addition of problem-specific constraints. Constraints are derived from background knowledge and specify information that may not be expressible in the traditional feature-value representation used for machine learning data sets. As noted by Beck and Fox (1998), constraints are important in search algorithms because they narrow the space of possible solutions and provide a direction for the search through that space. In this way, constraints can refine the representational bias of the algorithm.

Constraints have been used successfully in several areas of supervised machine learning, either in a preprocessing step (Hirsh and Noordewier 1994, Feldman and Hirsh 1996) or as modifications to the search operators (Janikow 1996, Lampinen and Selonen 1997). We focus here on describing work that makes use of background knowledge as constraints in clustering algorithms. Although the problem of constrained clustering has been treated extensively in the statistics literature, only limited work has been done on applying such methods in the machine learning field. In addition, the statistics community has been concerned almost exclusively with batch algorithms and a restricted class of constraints.

4.1 Constrained Clustering in Statistical Data Analysis

A clustering task can be phrased as an optimization problem:[2] the goal is to select a partition P from the set of *feasible* partitions $\Phi(D)$ such that $f(P)$ is minimized (or maximized), where f is the objective function.[3] Feasible partitions are clusterings of the data set D that satisfy requirements imposed by the algorithm (note, therefore, that $\Phi(D)$ is determined by the algorithm's bias). Thus, $\Phi(D)$ serves to exclude invalid hypotheses, such as clusterings with overlapping clusters or partitions that do not exhaust D. This notion can be extended so that $\Phi(D)$ also incorporates problem-specific constraints, thereby enabling the further exclusion of unpromising areas of the search space.

Most work on refining $\Phi(D)$ to date has been in *contiguity-constrained clustering* (Gordon 1973, Murtagh 1985, Gordon 1996, Bachar and Lerman 1998), also known as *relational clustering* (Ferligoj and Batagelj

[2]The problem statement presented here is adapted from Batagelj and Ferligoj (1998).

[3]Without loss of generality, we will restrict this discussion to partitions. Feasible hierarchies are those composed of feasible partitions.

1982, Ferligoj and Batagelj 1983) or *conditional clustering* (Lefkovitch 1980). In this problem formulation, additional information is known about the structure of the desired solution that takes the form of a "neighborhood" or "contiguity" relation.[4] For example, knowing which pixels are neighbors is useful information when doing image segmentation, since objects are usually contiguous. Figure 2 illustrates a simple example. The image in part (a) is composed of 25 pixels and has three distinct regions. The corresponding contiguity relation R forms a connected graph $(D,\ R)$ as shown in part (b) where each node represents an instance (pixel) and each edge indicates that the pair of instances is contiguous. In this case, "contiguous" pixels are those which share a side (i.e. the north, east, south, and west neighbors); it is also possible to use a definition that includes all touching pixels (i.e. the diagonal neighbors as well). If, as in this example, the goal is simply to express the existence (or absence) of a relationship, then the edges are unweighted; if varying degrees of contiguity are required, then they can be weighted. The resulting clusters, in this formulation, must preserve the structure given in R. $\Phi(D)$ is thus restricted so that only solutions which satisfy R are allowed, i.e. for each clustering $P \in \Phi(D)$, each $C \in P$ must be a valid subgraph $(C,\ R \cap C \times C)$. This is accomplished in hierarchical algorithms by restricting which cluster pairs can be considered for merging at each step; Batagelj and Ferligoj (1998) describe a method for satisfying relation R in an iterative relocation algorithm.

Relation R is therefore treated as a set of *soft* constraints; not all links in R must be preserved in the final result. However, the absence of a link between two elements of $C \times C$ is a *hard* constraint. If R in the image segmentation example included a link between pixels 21 and 5, a two-cluster solution suddenly would be possible; otherwise, the two dark regions cannot be merged without also merging the intervening white pixels.

Contiguity-constrained clustering requires that the relation R be complete, i.e. the induced graph is connected. It is common to handle the disconnected case by separately analyzing each connected component, since as above, it is assumed that the lack of a link between two components prevents them from being clustered together. While this may in some cases be a valid assumption (as in the neighborhood relation for image segmentation), for general relations it will not be. Incomplete knowledge of the relation R is one possibility. In addition, some relations may only be defined over a subset of the instances. Ambroise et al. (1997) introduce an EM-based method for contiguity-constrained clustering that does allow the merging of clusters that are not adjacent in terms of R. This is accomplished by adding a weighted penalty to the likelihood function that accounts for how many of the neighborhood relationships are violated (previous methods could be viewed as using an infinite weight for this penalty). The algorithm is, however, very sensitive to the choice of weighting factor β for the penalty.

Ferligoj and Batagelj (1983) extend the notion of relational constraints to apply to asymmetric relations. However, their method appears to apply only to transitive relations since it computes a transitive closure of R. More importantly, they restrict their attention to relations that indicate "sameness"; their method fails, for example, if $(x, y) \in R$ means that x and y *cannot* be clustered together (a valid relation, but one that does not indicate "sameness").

In summary, the methods which have been explored in the statistics community have generally focused on batch algorithms and contiguity constraints. While these methods are appropriate for some problems, they are inadequate for situations where an online solution is desired or the given constraints do not satisfy the assumptions of completeness, transitivity, and "sameness".

4.2 Constrained Clustering in Unsupervised Machine Learning

Some limited work has been done in constrained unsupervised machine learning, primarily with hierarchical methods. However, it has in general relied upon either artificial constraints or user intervention for success. In addition, only a limited set of constraints has been investigated. Most importantly, a meaningful method for evaluation has been lacking.

Thompson and Langley (1992) investigated the use of soft constraints with incremental hierarchical algorithms. Rather than a contiguity relation R, the algorithms were presented with an initial "priming" concept hierarchy which they were later free to modify. While the authors did demonstrate that predictive accuracy improved, it is not clear how to apply this technique to other problems or data sets. For two of the

[4] Murtagh (1995a, 1995b) has shown that Kohonen maps (also known as self-organizing maps) naturally produce output that contains such contiguity constraints. The cluster centers that are formed are usually laid out in a two-dimensional grid, and subsequent clustering of the output should take into account this contiguity information.

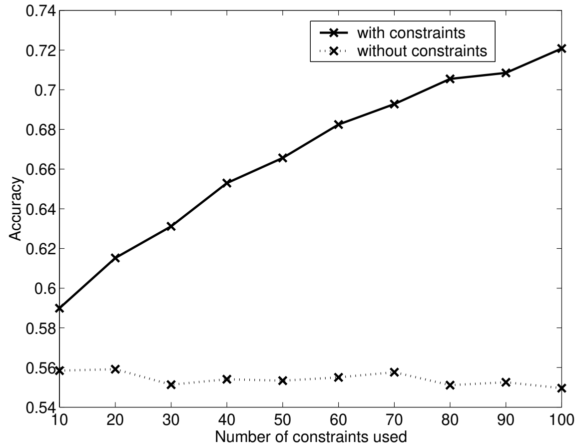


Figure 3: Performance on part of speech data

three algorithms, the initial hierarchies used were manually constructed to fit conceptual clusters present in the data, which was artificially generated. In the third case, the authors report that the data set used was too small to conclusively demonstrate accuracy improvements.

A constrained batch hierarchical algorithm was developed by Béjar and Cortés (1998). This algorithm is able to incorporate constraints in the form of rules that test for specific attribute values and predict the appropriate class. Incomplete knowledge can therefore be incorporated, but only in the specified form. Talavera and Béjar (1999) developed a similar algorithm which expands slightly on the work of Béjar and Cortés in that it is possible to specify that a set of instances should be grouped together even when the appropriate class label is not known. However, this system relies on user input to operate: the user selects which level of the hierarchy to use as the derived partition. Thus, the user determines, perhaps inadvertently, which constraints are actually reflected in the result. If the partition is chosen below the point where a set of constrained instances are merged together, then this constraint will not be satisfied. In both systems, the constraint framework does not provide a way to *prevent* two items from being grouped together, thus limiting what can be expressed.

An excellent example of incorporating domain knowledge as constraints is given by Pereira and Schabes (1992). They approach the problem of inferring a context-free grammar by using a consistency constraint on where constituent boundaries can fall. Their algorithm is an extension of the inside-outside algorithm, which possesses an EM flavor, and once again uses a penalty to disallow invalid hypotheses. In contrast to the work of Ambroise et al., their penalty is discrete (0 or 1) rather than continuous. Although this work does not make use of a clustering algorithm, it is an example of a technique that could be useful in our own investigations.

A chronic drawback of all of these methods is that there is no measurement of how much information is contained in the knowledge presented to each algorithm. When providing a set of valid constraints to an algorithm, an accuracy improvement should be unavoidable. The magnitude of this improvement, however, is of interest: if accuracy is higher simply because the algorithm is regurgitating knowledge that was previously fed to it, then it is not clear that a learning improvement has been made. We have addressed this issue in our clustering work by using an evaluation method that identifies which accuracy improvements are, in fact, due to a learning effect (Wagstaff and Cardie 2000).

5 Work to Date

To date, we have investigated the incorporation of instance-level hard constraints into a single clustering algorithm (a partitioning variation of COBWEB (Fisher 1987)). The resulting algorithm, which enforces all specified constraints, is called COP-COBWEB (for "constraint-partitioning"). We found that clustering accuracy improved dramatically with the addition of relatively few constraints (Wagstaff and Cardie 2000). Held-out accuracy (i.e. accuracy on instances not involved in the specified constraints) improved up to 11% after incorporating 50 randomly selected constraints, and up to 17% with 100 constraints (see Figure 3 for an example). We believe that intelligent selection of constraints could be of even more benefit, since some of the specified constraints indicated information that the algorithm was able to deduce on its own. In addition to the increases in accuracy, we observed impressive decreases in runtime due to the reduction in the size of the search space.[5] The constraints therefore strengthened the representational bias without sacrificing correctness.

We investigated two kinds of constraints: *must-link* and *cannot-link* constraints, both of which are hard constraints and operate at the instance level. A must-link constraint specifies that two instances must be in the same cluster; a cannot-link constraint indicates that they must be in different clusters. We therefore incorporated two relations, neither of which was assumed to be complete. The must-link relation is transitive, while the cannot-link relation is not.

Due to the success of this initial investigation, we propose the development and evaluation of new methods that integrate unsupervised learning techniques with background knowledge. The goal is to increase accuracy and to improve efficiency. In the rest of this paper, we first outline our main areas of interest and then describe our intended methods for evaluation, finishing with a summary of the research issues central to this proposal.

6 Next Steps

Our overall goal is to develop constraint-capable versions of each type of clustering algorithm identified in Section 3. We here lay out our master plan for further progress on the topic and identify a sample data set for each each algorithm we plan to test. We have already worked with an incremental partitioning algorithm (COP-COBWEB), but we have only evaluated it using randomly-generated constraints and four data sets: a part-of-speech tagging data set and three others from the UCI repository (Blake and Merz 1998). We therefore plan to apply this algorithm to a significant real-world problem (Section 6.1) that presents us with the chance to incorporate realistic constraints derived from background knowledge. Next, we will focus in turn on each of the remaining categories of clustering algorithms, applying the constraint-capable versions to appropriate real-world problems (Sections 6.2 and 6.3). These data sets are representative but not all-encompassing; we also expect to make use of other data sets. A summary of our plans is shown in the following table.

Algorithm type	Algorithm	Data set	Proposed constraints
Batch partitioning	k-means (EM)	GPS traces	Width of lanes, restricted contiguity constraints
Batch hierarchical	Agglomerative	SWISS-PROT	Existing labels, subsequence matches
Incremental partitioning	COP-COBWEB	MUC-6, MUC-7	Linguistic coreference constraints

An entry for an incremental hierarchical algorithm is notably missing from this table. We plan to use COBWEB for this category but have not yet identified a specific problem domain. We expect to identify such a problem in the near future.

We are secondarily interested in a comparison of constrained clustering performance on partially labeled data sets to that of supervised or weakly-supervised algorithms (Section 6.4). In addition, we hope to examine behavior on artificial data sets to understand specific strengths and weaknesses of constraint approaches (Section 6.5).

6.1 Noun Phrase Coreference

Noun phrase coreference is an important problem in the domain of natural language processing. In this task, the goal is to identify which noun phrases are coreferent, i.e. which noun phrases refer to the same real-world entity. The problem can be viewed as a clustering task (Cardie and Wagstaff 1999), where the noun phrases from a single document are partitioned into a set of clusters so that there is one cluster for each

[5]Interestingly, in addition to constraints improving the efficiency of clustering algorithms, it has been shown that clustering can improve the efficiency of traditional constraint satisfaction methods (Ellman 1993).

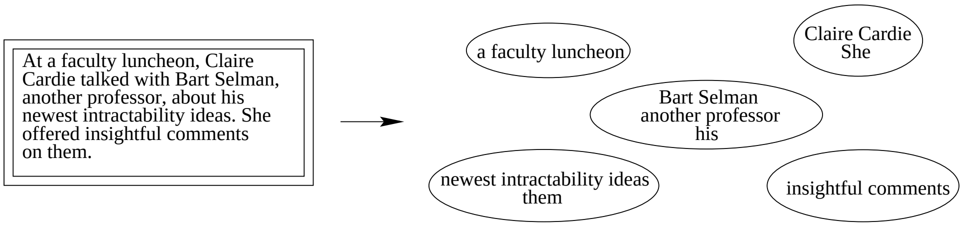


Figure 4: Noun Phrase Coreference

entity mentioned in the text. The document in Figure 4, for example, mentions several distinct entities. One "coreference" cluster ultimately should contain all noun phrases that refer to Bart Selman ("Bart Selman", "another professor", and "his"), while another cluster should contain all noun phrases that refer to Claire Cardie ("Claire Cardie" and "She").

The constraints. Due to our past work on this problem, we believe that it represents a domain likely to benefit from the addition of constraints. Semantic and syntactic linguistic knowledge allows us to determine *a priori* that certain pairs of noun phrases either must be, or cannot be, in the same coreference cluster. For example, if two noun phrases are in an appositive relationship, then they must[6] be coreferent, regardless of their other features. Likewise, we know that "She" and "his" cannot be coreferent, since they mismatch on gender. Once again, this is the case regardless of their other features (e.g., they both refer to humans, they are close together in the document, etc.). As we discovered in our earlier work, constructing a custom distance measure that can accommodate a variety of such preferences and restrictions is non-trivial and time consuming (and as noted above, algorithms that accommodate preferences usually require that the relation be defined over all instances, and none of the algorithms surveyed can handle a relation that prevents merging). We posit that the desired relationships can be more naturally expressed as a set of constraints that a constraint-handling algorithm can incorporate. We intend to discover which kinds of constraints most effectively express background knowledge in this domain.

Evaluation. Noun phrase coreference has been extensively studied by the participants of the recent Message Understanding Conferences (MUC-6 1995, MUC-7 1998). Consequently, we have several corpora that can be used for development and evaluation. Evaluation for the coreference task can be done in at least two ways. One method is to provide as input the exact noun phrases given in the answer key and compute a clustering accuracy score from the results we obtain. This gives us a pure measure of our algorithm's ability to detect coreference relations. Alternatively, we can use as input all of the document's noun phrases as computed by a system such as that described by Cardie and Pierce (1998). We would then apply the MUC coreference scorer (Vilain et al. 1995) to calculate accuracy as a combination of recall and precision. This allows us to evaluate performance in a more realistic situation, where the "true" noun phrases are not known and must be extracted from the text. In addition, these accuracy figures can be compared directly to the results reported by the MUC-6 and MUC-7 participants, who were also required to extract the noun phrases as input to their coreference systems.

The only other learning technique that has been applied to the general problem of noun phrase coreference[7] makes use of decision trees (McCarthy and Lehnert 1995). This is a supervised approach that learns from pairs of noun phrases that have been marked as "coreferent" or "not coreferent". They reported an accuracy of 47% on the MUC-6 data. Our previous clustering approach to the problem yielded 54% accuracy. Of the systems participating in the MUC-6 evaluation, the highest accuracy attained was 65% (by a system that made use of hand-crafted heuristics), indicating that there is clearly much room left for improvement.

Table 1: Partial SWISS-PROT entry for protein P04217, a human glycoprotein

Attribute	Value
ID	A1BG-HUMAN STANDARD; PRT; 474 AA.
DE	ALPHA-1B-GLYCOPROTEIN.
OS	Homo sapiens (Human).
CC	-!- FUNCTION: PLASMA GLYCOPROTEIN OF UNKNOWN FUNCTION.
SQ	AIFYETQPSL WAESESLLKP LANVTLTCQA RLETPDFQLF KNGVAQEPVH LDSPAIKHQF ...

6.2 Protein Function Assignment

Other good candidates for evaluating constraint techniques are domains where class labels are known for some, but not all, of the instances. One example is the SWISS-PROT protein database (Bairoch and Boeckman 1992). The functions of some proteins are known, but others have yet to be discovered. The goal is to place the proteins in groups so that proteins with similar functions are grouped together (even if the exact function is not known). As an example, Table 1 contains the SWISS-PROT entry for protein P04217, a human glycoprotein whose function is unknown. Values for the protein's ID, description (DE), originating species (OS), comments (CC), and sequence (SQ) are shown; the full entry contains several more fields that have been omitted here.

The constraints. Yona et al. (1999) indicates that 15 complete genomes are known, but between 35 and 50% of their proteins have not yet been assigned a function. While not enough information is available for purely supervised methods, a constraint-clustering approach to the problem could use the incomplete class information to constrain the placement of instances into the proper "protein function" cluster. Once again, previous approaches require that the constraint relation include all instances and cannot handle this sort of incomplete information. The work of Béjar and Cortés (1998) and Talavera and Béjar (1999) does allow for incomplete information, but requires that it can be specified as a conjunction of attribute-value tests, which may not be possible here.

Yona et al. applied an agglomerative clustering algorithm to this problem by seeking strongly connected components in a graph that represents the protein database. Each node in the graph represents a protein sequence; each edge carries a weight that corresponds to the similarity of its two endpoints. This completely unsupervised method is an attempt to automatically discover protein family groupings. We propose modifying this algorithm to take advantage of the currently-known information to ensure the correct placement of the proteins that are known, thus improving the chance of placing an unknown protein correctly.

In addition, we need not rely solely on the labeled examples for constraints. Our general approach is concerned with domain knowledge, and the above constraints would ideally be augmented with information that could go beyond the known labels. For instance, these proteins are commonly represented by their amino acid sequences (see the SQ field in Table 1). Matches between two proteins on subsequences is usually, but not always, indicative of similarity. Information on how reliable the various subsequence matches are could be converted into probabilistic constraints, e.g., $contains - subseq(p_i, S_k) \wedge contains - subseq(p_j, S_k) \Rightarrow P(class(p_i) == class(p_j)) = 0.5$, where p_i and p_j are specific proteins and S_k is a common subsequence.

Evaluation. The SWISS-PROT database is constantly updated as new findings are announced. One method for evaluating our algorithm would be to take an older version of the database (where fewer protein functions are known) and then make predictions for proteins whose functions have since been discovered. In addition, we would like to use the latest database and compare our method's predictions on the unknown proteins to that of a supervised method that treats the unknown ones as a test set. Finally, we can adopt the approach used by Yona et al., which is to compare the output to that of protein databases that have been constructed based on other criteria (e.g., motifs rather than sequences).

[6]Although these examples appear to be most appropriately expressed as hard constraints, natural language is full of exceptions to any rule. Therefore, soft constraints may be more appropriate.

[7]Ge et al. (1998) made use of a statistical supervised approach, but it was restricted to pronoun resolution.

6.3 GPS Mining for Lane Detection

Rogers et al. (1999) have been working on the problem of mining GPS traces to improve digital maps down to the scale of individual lanes. Because people tend to drive within lanes rather than on the lane boundaries, GPS data passively collected from human drivers can be clustered to locate the lanes automatically. Rogers et al. view this as a one-dimensional clustering problem (the only attribute is the lateral offset from the road centerline). They use an agglomerative algorithm, terminating the process when the two closest clusters are further apart than the minimum lane width.

The constraints. In upcoming months, we will be working with Rogers et al. to improve this basic system by introducing background knowledge in the form of constraints. For example, it should be possible to also make use of a maximum lane width to eliminate some merges from being considered. We are also interested in incorporating some of the inherent structure of the problem: although the clustering is done in one dimension, the fact that two instances came from the same GPS trace is also useful information, in that they should generally be assigned to the same lane (but not always, since drivers do make lane changes). Incorporating this kind of information presents an interesting challenge and should also help to compensate for the noise found in the GPS data.

Although the current system uses an agglomerative method, we plan (in conjunction with Rogers et al.) to develop an alternative that uses an EM-based method. This will present an excellent opportunity to investigate methods for incorporating constraints for EM algorithms aside from penalty-based approaches.

Evaluation. As noted by Rogers et al., evaluation in this domain is difficult because, as in many unsupervised applications, the "truth" is simply not known. In order to produce accuracy results, the authors manually tagged the data and compared the clustering output to those tags. This can be done for small evaluations, although as noted above, it can be infeasible for large data sets.

6.4 Comparison to Supervised Methods

The domains described in the previous section can all be described as clustering problems: the goal is to divide the data into meaningful groups. Correspondingly, solutions are generally evaluated in terms of their ability to correctly discover structure in the given data set. However, another important class of problems focuses on the task of *prediction* (or classification), where evaluation is commonly measured in terms of predictive accuracy on a held-out test set. Both supervised and unsupervised methods can be applied to prediction tasks. In the supervised case, the algorithm uses a labeled data set to build a representation of the concept induced by the data and then uses that representation to predict the labels of unseen data. In the unsupervised case, the algorithm must construct a representation from unlabeled data and then predict where in the partition or hierarchy each new instance belongs (it cannot actually label the new instance, since it doesn't have label information for the training data).

Since supervised approaches have access to information about the precise concept to be learned (i.e. the data is labeled), we might expect their performance to be generally higher than that of unsupervised algorithms, which operate with much less specific assistance. However, as noted above, creating supervisory information may be difficult, expensive, or impossible. Thus, if we can devise methods which require less information but still perform at or near the level of completely supervised approaches, we can reduce the labor demands and enable the application of machine learning techniques to a broader range of domains.

This goal is reminiscent of recent work done in *weakly-supervised* learning, e.g., active learning (Cohn et al. 1994), bootstrapping (Riloff and Jones 1999, Yarowsky 1995), and co-training (Blum and Mitchell 1998)). These methods aim to achieve performance that is comparable to a completely supervised method while reducing the required number of labeled examples. When making use of partially labeled data, our method can also be considered weakly-supervised, and we intend to conduct experiments that will allow us to compare constrained clustering to other weakly-supervised methods.

We therefore plan to make use of data sets that are typically used in conjunction with supervised methods (e.g., the UCI data sets) and compare the predictive performance of constraint-capable algorithms to that of supervised methods. In addition, we would like to compare our algorithms to the weakly-supervised ones mentioned above. As an example, we can experiment with creating constraints based on knowing the labels of random subsets of the data (akin to the partially-labeled case above). Ideally, we would like to be able to make claims such as, "If at least $x\%$ of the data is correctly labeled, we can achieve performance comparable to completely supervised methods." The underlying hypothesis is that some labels are, in fact, redundant information, in that they can be rederived by a properly biased learning algorithm. If relationships between instances can be deduced on the basis of their features alone (an unsupervised approach), then we need not go to the trouble of labeling them. Our approach, then, will attempt to exploit this redundancy where possible.

6.5 Artificial Data Sets

Artificial data sets will be useful for exploring the behavior of constraint-capable algorithms in controlled situations. For example, in experiments with a data set that had a lopsided distribution of instances, we noticed lower performance both with and without constraints. Minority class representation is a pervasive problem in real data (Cardie and Howe 1997), and we would like to understand how sensitive our approach may be to those effects. Generally, performance is much lower on underrepresented classes. The minority classes are represented by relatively few items, thus suffering from a paucity of information. It is therefore possible that the additional information conveyed by the constraints can be of even more assistance for decisions involving instances from those classes. To test these theories, we can experiment with artificial data sets that have specific class distributions.

We also expect to use artificial data to investigate a surprising phenomenon we observed in our preliminary experiments with COP-COBWEB. As noted above, the objective function used by this algorithm attempts to maximize the category utility (CU) of the resulting partition. Contrary to expectations, we observed that the CU of the "correct" (fully-constrained) partition was *lower* than that of the "less correct" one obtained without using constraints. We suspect that the discrepancy between the two CU values is higher in situations where there is less correlation between the class label and the other attributes. In some sense, this "CU gap" could be an indicator of the relative difficulty of the data set with respect to learning the concept defined by its labels. More generally, we are also curious as to whether the same effect is observed with other objective functions. Once again, artificial data sets will allow us to investigate this issue by controlling for the correlations between attribute values and class labels.

7 Evaluation of Constraint Techniques

There are two common methods for evaluating clustering algorithms in the machine learning literature. When the result of the clustering algorithm will be used to classify new instances, it is common to evaluate performance on a held-out test set. If the goal is simply to recover structure found in the data set, a comparison to the supervised labels is often made. To compare two hierarchies, it is possible to construct a similarity matrix S for each such that S_{ij} is the level in the hierarchy where instances i and j were merged. For partitioning algorithms, the *majority vote* metric is sometimes used: each cluster is tagged with the class that most commonly occurs among its members, and any instances with different labels are considered errors. Unfortunately, this metric permits 100% accuracy trivially when each instance is placed in its own cluster. In essence, it is more a measure of cluster "purity" than an evaluation of how closely a given solution matches the true solution.

Consequently, we turn to other methods for evaluating the agreement between two partitions. Rand (1971) proposed a method which, given partitions P_1 and P_2 on data set D, considers each pairwise relationship between two instances D_i and D_j and calculates whether P_1 and P_2 agree or not. This is the approach we adopted for our past work, modified to only consider relationships that involve instances not directly (or transitively) affected by the given constraints.

Carletta (1996) has suggested the use of the kappa statistic (Siegel and Castellan, Jr. 1988) to determine the amount of agreement between multiple coders on classification tasks. The kappa statistic calculates the raw agreement of the coders $P(A)$ and then discounts this amount to compensate for how much agreement would be expected by chance $P(E)$, in the following way:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

This approach can similarly be used to compare the output of multiple clustering algorithms and/or label sets. It is also possible to determine how significant a reported agreement value is.

Additionally, Gordon (1981) suggests the use of *admissibility criteria*. Clustering algorithms (or their output) can be characterized in terms of which criteria they satisfy. For example, an algorithm is *point proportion admissible* if adding multiple copies of the same instance to the data set does not result in a modified clustering. Similarly, an algorithm is *cluster omission admissible* if, after removing all instances that fall into a specified cluster, the result of re-clustering contains exactly the $k-1$ clusters from the original output. Knowledge about the characteristics of a given algorithm provides a method for determining whether it is appropriate for a specific problem.

In general, the evaluation of clustering algorithms is always a challenge. An unsupervised algorithm by definition operates without access to a "true" answer. It is possible, as detailed above, to apply such algorithms to labeled data sets and afterwards compare the output to the labels, but this can be deceptive. It only measures the algorithm's ability to recreate the concept responsible for the labeling of the data, which may not coincide precisely with patterns actually derivable from the given features. Therefore, there is no indication of how well the algorithm detects patterns that are actually present in the data but not reflected in the labels. Often, several meaningful but distinct clusterings are possible (Anderberg 1973). One of the strengths of unsupervised approaches is the identification of significant patterns, including in some cases patterns that humans have not yet recognized (e.g., Autoclass discovered a new class of stars from infra-red spectra (Cheeseman et al. 1989)). For this reason, clustering is often referred to as an *exploratory* data analysis tool (Anderberg 1973). Nevertheless, it is important to obtain some measure of the "goodness" of the results of a clustering algorithm, and the above methods are a step in the right direction.

8 Conclusion

Our ultimate goal is to show that constraints can be of assistance for a variety of clustering algorithms. Initial work will include experimenting with an algorithm of each type identified in Section 3. We have already done work with an incremental partitioning algorithm, which leaves the other three as open areas of research. Each algorithm type may require a different constraint formulation: for example, hierarchical algorithms demand a method for expressing hierarchical constraints (as opposed to the instance-level hard constraints that were suitable for COP-COBWEB).

We also plan to investigate methods for generating constraints to be used by these techniques. Clearly, for a problem to benefit from a constraint approach, it must be associated with some amount of background knowledge. Methods for collecting this knowledge might range from consulting domain experts to "mining" documentation that accompanies a data set. In addition, the knowledge may need to be re-expressed in order to be useful to a constraint-capable algorithm.

We have identified an area of machine learning that, according to our preliminary work, stands to benefit greatly from an investigation into the use of background knowledge as constraints. Although work has been done with restricted forms of relational constraints, our proposal encompasses a much broader set of algorithms as well as more expressive constraints. We are also interested in related issues such as minority class effects, the observed CU discrepancy, and how to generate useful constraints. We expect this work to contribute to the field of machine learning by providing a means for enhancing unsupervised techniques, which have greater applicability than their supervised counterparts, with relevant background information, which can refine bias and serve to compensate for incomplete feature sets.

Acknowledgements

This work was supported in part by an NSF Graduate Fellowship and NSF Grant IRI-9624639. We are deeply grateful to Claire Cardie for her advising, assistance, input, and guidance. We would like to thank the entire thesis committee (Jim Bell, Claire Cardie, and Bart Selman) and Lillian Lee for insightful suggestions and reference recommendations. In addition, we thank Westley Weimer for proofreading this document and for assistance as the research progressed.

References

AAAI-92 Workshop. 1992. *Proceedings of the AAAI-92 Workshop on Constraining Learning with Prior Knowledge*. The AAAI Press, San Mateo, CA.

R. Agrawal, T. Imielinski, and A. Swami. 1993. Mining associations between sets of items in massive databases. In *Proceedings of the ACM-SIGMOD 1993 International Conference on Management of Data*, pages 207–216, Washington, D.C.

C. Ambroise, M. Dang, and G. Govaert. 1997. Clustering of spatial data by the EM algorithm. In A. Soares et al., editor, *geoENV I-Geostatistics for Environmental Applications*, pages 493–504. Kluwer Academic Publishers.

M. Anderberg. 1973. *Cluster Analysis for Applications*. Academic Press.

K. Bachar and I. Lerman. 1998. Statistical conditions for a linear complexity for an algorithm of hierarchical classification under constraint of contiguity. In A. Rizzi, M. Vichi, and H. Bock, editors, *Advances in Data Science and Classification*, pages 131–136. Springer-Verlag.

A. Bairoch and B. Boeckman. 1992. The SWISS-PROT protein sequence data bank. *Nucleic Acids Research*, 20:2019–2022.

V. Batagelj and A. Ferligoj. 1998. Constrained clustering problems. In A. Rizzi, M. Vichi, and H. Bock, editors, *Advances in Data Science and Classification*, pages 137–144. Springer-Verlag.

J. C. Beck and M. S. Fox. 1998. A generic framework for constraint-directed search and scheduling. *AI Magazine*, 19(4):101–130.

J. Béjar and U. Cortés. 1998. Experiments with domain knowledge in unsupervised learning: Using and revising theories. *Computación y Sistemas*, 1(3):136–143.

C. L. Blake and C. J. Merz. 1998. UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html, University of California, Irvine, Department of Information and Computer Sciences.

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pages 92–100.

C. E. Brodley. 1994. *Recursive Automatic Algorithm Selection for Inductive Learning*. Ph.D. thesis, University of Massachusetts, Amherst, MA.

C. Cardie and N. Howe. 1997. Improving minority class prediction using case-specific feature weights. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 57–65, Vanderbilt University, Memphis, TN. Morgan Kaufmann.

C. Cardie and D. Pierce. 1998. Error-driven pruning of treebank grammars for base noun phrase identification. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and COLING-98*, pages 218–224, University of Montreal, Montreal, Canada. Association for Computational Linguistics.

C. Cardie and K. Wagstaff. 1999. Noun phrase coreference as clustering. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 82–89, University of Maryland, MD. Association for Computational Linguistics.

J. Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics*, 22(2):249–254.

P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. 1988. Autoclass: A Bayesian classification system. In *Proceedings of the Fifth International Workshop on Machine Learning*, pages 54–64, Ann Arbor, MI. Morgan Kaufmann.

P. Cheeseman, J. Stutz, M. Self, W. Taylor, J. Geobel, K. Volk, and H. Walker. 1989. Automatic classification of spectra from the infrared astronomical satellite (IRAS). NASA Reference Publication 1217, National Technical Information Service, Springfield, VA.

D. Cohn, L. Atlas, and R. Ladner. 1994. Improving generalization with active learning. *Machine Learning*, 15(2):201–221.

L. De Raedt, M. Bruynooghe, and B. Martens. 1991. Integrity constraints and interactive concept-learning. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 394–398. Morgan Kaufmann.

A. P. Dempster, N. M Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, Series B.

T. Ellman. 1993. Synthesis of abstraction hierarchies for constraint satisfaction by clustering approximately equivalent objects. In *Proceedings of the Tenth International Conference on Machine Learning*, University of Massachusetts, Amherst, MA.

R. Feldman and H. Hirsh. 1996. Mining associations in text in the presence of background knowledge. In *Proceedings of the Second International Conference on Knowledge Discovery from Databases*.

A. Ferligoj and V. Batagelj. 1982. Clustering with relational constraint. *Psychometrika*, 47(4):413–426.

A. Ferligoj and V. Batagelj. 1983. Some types of clustering with relational constraints. *Psychometrika*, 48(4):541–552.

D. Fisher. 1987. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172.

E. W. Forgy. 1965. *Cluster Analysis of Multivariate Data: Efficiency Versus Interpretability of Classifications*. Biometric Society Meetings, Riverside, CA.

N. Ge, J. Hale, and E. Charniak. 1998. A statistical approach to anaphora resolution. In E. Charniak, editor, *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 161–170, Montreal, Canada. ACL SIGDAT.

M. A. Gluck and J. E. Corter. 1985. Information, uncertainty, and the utility of categories. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pages 283–287, Hillsdale, NJ. Lawrence Erlbaum.

D. Gordon and M. desJardins. 1995. Evaluation and selection of biases in machine learning. *Machine Learning*, 20(1/2):5–22.

A. D. Gordon. 1973. Classification in the presence of constraints. *Biometrics*, 29:821–827.

A. D. Gordon. 1981. *Classification*. Chapman and Hall, New York, NY.

A. D. Gordon. 1996. A survey of constrained classification. *Computational Statistics & Data Analysis*, 21:17–29.

J. A. Hartigan. 1975. *Clustering algorithms*. Wiley, New York, NY.

H. Hirsh and M. Noordewier. 1994. Using background knowledge to improve inductive learning of DNA sequences. In *Proceedings of the Tenth IEEE Conference on Artificial Intelligence for Applications*, pages 351–357. IEEE Computer Society Press.

R. C. Jancey. 1966. Multidimensional group analysis. *Australian Journal of Botany*, 14(1):127–130.

C. Janikow. 1996. A methodology for processing problem constraints in genetic programming. *Computers and Mathematics with Applications*, 32(8):97–113.

M. Křivánek. 1991. *Algorithmic and Geometric Aspects of Cluster Analysis*. Academia Nakladatelství Československé, Praha.

J. Lampinen and A. Selonen. 1997. Using background knowledge in multilayer perceptron learning. In M. Frydrych, J. Parkkinen, and A. Visa, editors, *Proceedings of the Tenth Scandinavian Conference on Image Analysis*, volume 2, pages 545–549.

G. N. Lance and W. T. Williams. 1967. A general theory of classificatory sorting strategies, 1. hierarchical systems. *The Computer Journal*, 9:373–380.

L. P. Lefkovitch. 1980. Conditional clustering. *Biometrics*, 36:43–58.

J. B. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Symposium on Math, Statistics, and Probability*, volume 1, pages 281–297, Berkeley, CA. University of California Press.

J. McCarthy and W. Lehnert. 1995. Using decision trees for coreference resolution. In C. Mellish, editor, *Proceedings of the Fourteenth International Conference on Artificial Intelligence*, pages 1050–1055.

T. Mitchell. 1997. *Machine Learning*. McGraw-Hill.

MUC-6. 1995. *Proceedings of the Sixth Message Understanding Conference*. Morgan Kaufmann, San Francisco, CA.

MUC-7. 1998. *Proceedings of the Seventh Message Understanding Conference*. Morgan Kaufmann, San Francisco, CA.

F. Murtagh. 1985. A survey of algorithms for contiguity-constrained clustering and related problems. *The Computer Journal*, 28(1):82–88.

F. Murtagh. 1995a. The Kohonen self-organizing map method: an assessment. *Journal of Classification*, 12:165–190.

F. Murtagh. 1995b. Unsupervised catalog classification. In R. A. Shaw, H. E. Payne, and J. J. E. Hayes, editors, *Astronomical Data Analysis Software and Systems IV*, volume 77 of *ASP Conference Series*, pages 264–267.

F. Pereira and Y. Schabes. 1992. Inside-Outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Newark, DE.

W. M. Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(366):846–850.

E. Riloff and R. Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 474–479, Orlando, FL. AAAI Press / MIT Press.

S. Rogers, P. Langley, and C. Wilson. 1999. Mining GPS data to augment road models. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 104–113, San Diego, CA. ACM Press.

S. Siegel and N. J. Castellan, Jr. 1988. *Nonparametric Statistics for the Behavioral Sciences.* McGraw-Hill, second edition.

W. Spears and D. Gordon. 1991. Adaptive strategy selection for concept learning. In *Proceedings of the First International Workshop on Multistrategy Learning*, pages 231–246, Harpers Ferry, WV.

L. Talavera and J. Béjar. 1999. Integrating declarative knowledge in hierarchical clustering tasks. In *Proceedings of the International Symposium on Intelligent Data Analysis*, pages 211–222, Amsterdam, The Netherlands. Springer-Verlag.

K. Thompson and P. Langley. 1992. Case studies in the use of background knowledge: incremental concept formation. In *Proceedings of the AAAI-92 Workshop on Constraining Learning with Prior Knowledge*, pages 60–68, San Mateo, CA. The AAAI Press.

P. Utgoff. 1986. *Machine Learning of Inductive Bias.* Kluwer Academic Publishers.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the Sixth Message Understanding Conference*, pages 45–52, San Francisco, CA. Morgan Kaufmann.

K. Wagstaff and C. Cardie. 2000. Clustering with instance-level constraints. In *Proceedings of the Seventeenth International Conference on Machine Learning. (to appear).*

J. H. Ward, Jr. 1963. Hierarchical groupings to optimise an objective function. *Journal of the American Statistical Association*, 58(301):236–244.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, MA.

G. Yona, N. Linial, and M. Linial. 1999. Protomap: Automatic classification of protein sequences, a hierarchy of protein families. In *Proteins: Structure, Function, and Genetics*, volume 37, pages 360–378.